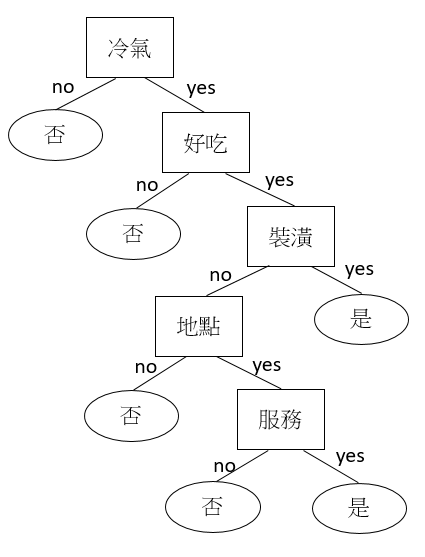
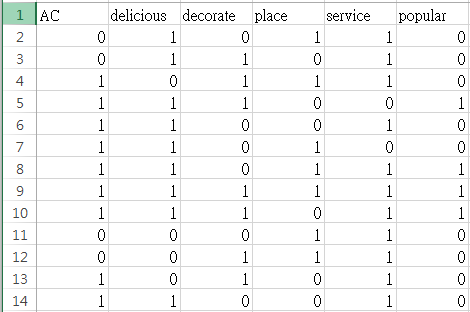
P78071549 陳珞安

Project2 : 餐廳是否受歡迎?

1. Decision Tree

My \_Model：



Data：

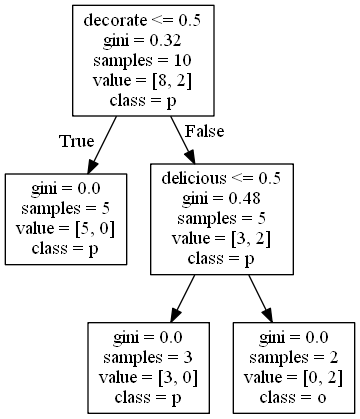
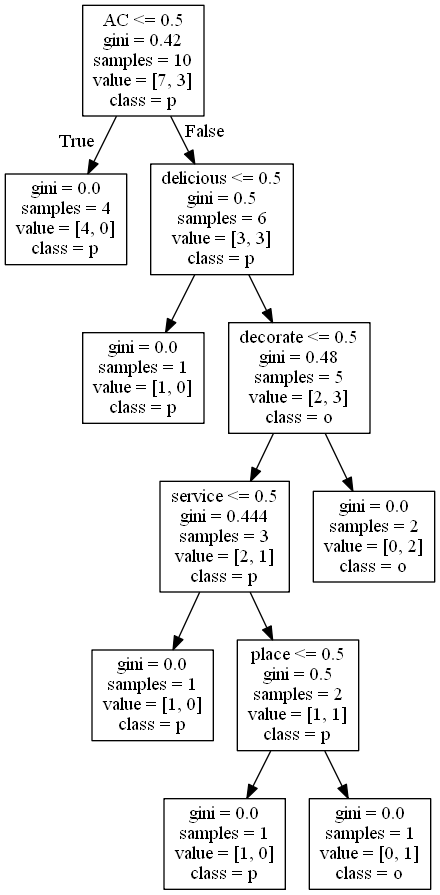
k(# of features) = 5 , M(# of datas) = 13

Result：

執行方式： python3 decision.py

執行結果：

因為程式是將80%的data作為training data，剩下的20%作為testing data(random)，由於自己產生的data量不足，所以只要沒有train到某幾條關鍵data，gini index算出來就會不一樣，跑出來的結果就會相差很多，圖(一)便是完全預測的，但圖(二)準確率卻只有0.3333，甚至也有機會跑出準確率為0的時候。



圖(二)準確率為0.333所畫出的decision tree

圖(一)準確率為1所畫出的decision tree

因此，便將data數(M)提升至30筆，這個時候大多的準確率都會落在0.8~1之間，我覺得原因是因為這個data的feature不多，而且都是二元的分類(是or否)，所以當正確的例子越多的時候，也代表相同情形的例子出現次數越多，這樣一來便比較不容易出現與預想相差極大的分類方式。

問題

這樣用random的方式所產生出的train\_set以及test\_set的優點是比較不會因為選擇某筆固定的data造成之後實驗的overfit或者是underfit，但缺點是在這邊無法固定train\_set，所以無法與其他分類的演算法做比較。

因此後來將資料做分割的時候，統一固定使用30筆的資料，並挑出其中的8、9、15、17、23、27作為test\_set，其他則做為train\_set來做實驗。

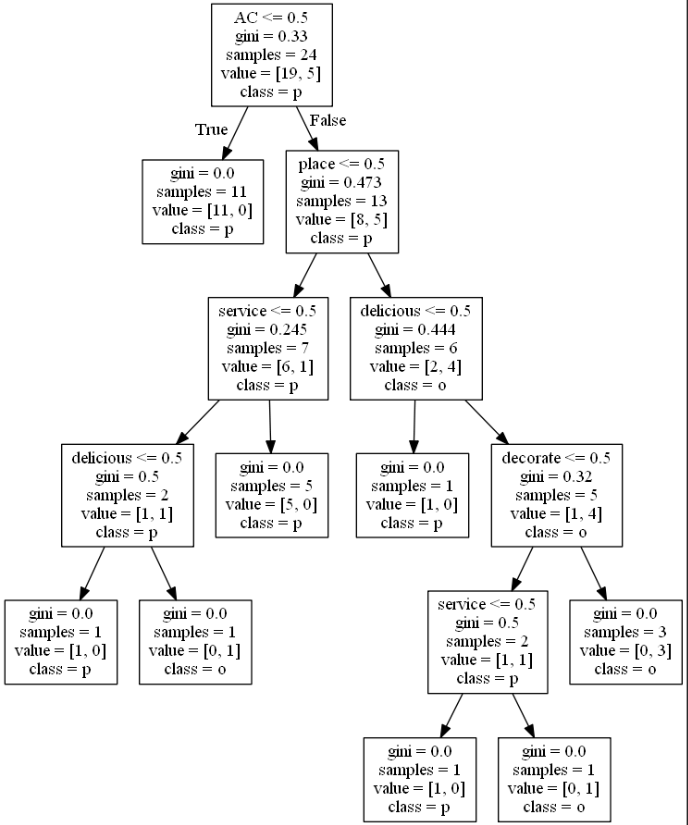


※統一random\_state(亂樹種子)，以達到之後都挑出同樣的train\_set

Result\_2：

執行方式： python3 decision.py

執行結果：



1. KNN

k(# of features) = 5 , M(# of datas) = 30

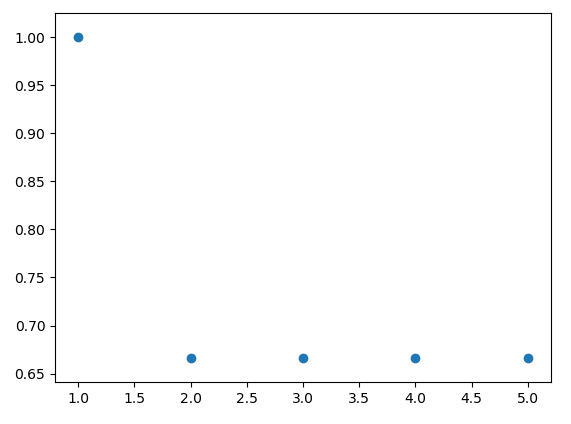
Result：

執行方式： python3 KNN.py

執行結果：

可以從下圖看出，在k=1的時候，準確率是1，但在之後卻都呈現0.68左右的準確率，可見在這個情況下K=1是準確率最高的。

(準確率)



(K值)

1. Gaussian Naïve Bayes

k(# of features) = 5 , M(# of datas) = 30

Result：

執行方式： python3 bayes.py

執行結果：

準確率： 0.833333333

1. Neural Network

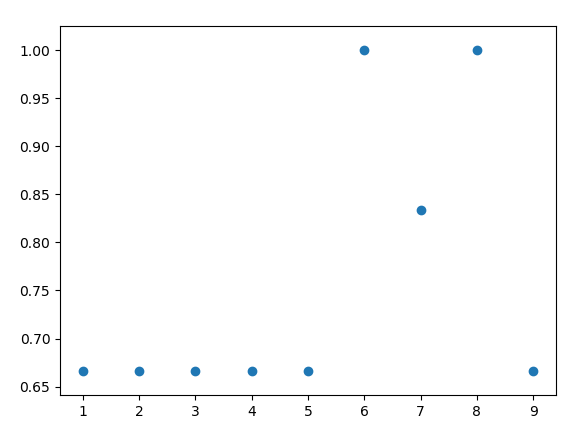
k(# of features) = 5 , M(# of datas) = 30

Result：

執行方式： python3 ANN.py

執行結果：

可以從下圖看出，在hidden layer數目為6以及8的時候準確率為百分之百，並且大致上維持的趨勢是hidden layer的數目越多，準確率通常越高。



(準確率)

(# of hidden layer)

1. Support Vector Machines

k(# of features) = 5 , M(# of datas) = 30

## Result：

執行方式： python3 SVM.py

執行結果：

準確率：0.6666666667

比較&結論:

在k=5,M=30以及train/test set都相同的情況下，準確率是ANN = KNN > Bayes > SVM > Decision Tree，雖然可能因為係數的調整而導致結果會大不相同，但基本上可以看出decision tree在分類的時候，優點是結構簡單並且沒有參數，但缺點是由於Generalization的能力太差，所以很難處理沒有train到的或者沒有出現過的值，也因此實際預測效果並不如其他的方法。

至於其他的方法如KNN、ANN……等等，因為大多有參數上的調整差距，往往需要根據每個不同的問題來調整最佳的參數，所以並不能一概的拿來比較準確率，但也多少能從個別演算法的參數調整來看出一些趨勢，例如類神經網路的黑盒運算，其hidden layers數目在多的時候通常預測結果會比少的時候還好。